Exercice 1

(D'après X-ENS 2016 MP-PC)

Une représentation simplifiée, réduite à deux tables de la base de donnée d'un réseau social est donnée dans la figure suivante :

`INDIVIDUS`

`id`	`nom`	`prenom`
1	Potter	Harry
2	Granger	Hermione
⋮	⋮	⋮

`LIENS`

`id1`	`id2`
1	2
2	1
⋮	⋮

La table `INDIVIDUS` répertorie les individus et contient les colonnes

- `id` (clé primaire), un entier identifiant chaque individu ;
- `nom`, une chaine de caractère donnant le nom de famille de l'individu ;
- `prenom`, une chaine de caractères donnant le prénom de l'individu.

La table `LIENS` répertorie les liens d'amitiés entre individus et contient les colonnes

- `id1`, entier identifiant le premier individu du lien d'amitié ;
- `id2`, entier identifiant le second individu du lien d'amitié ;

Par exemple Harry Potter et Hermione Granger sont amis dans le réseau social. On supposera par ailleurs que pour tout couple `(x,y)` présent dans la table `LIENS`, le couple `(y,x)` y est aussi présent.

1. Écrire une requête SQL qui renvoie les identifiants des amis de l'individu d'identifiant `x`.
2. Écrire une requête SQL qui renvoie les (noms,prénoms) des amis de l'individu d'identifiant `x`.
3. Écrire une requête SQL qui renvoie les identifiants des individus qui sont amis avec au moins un ami de l'individu d'identifiant `x`.

Exercice 2

(D'après Centrale-Supélec 2016 MP-PC-PSI-TSI)

Afin d'éviter les collisions entre avions, les altitudes de vol en croisière sont normalisées. Dans la majorité des pays, les avions volent à une altitude multiple de 1000 pieds (un pied vaut 30,48 cm) au-dessus de la surface isobare à 1013,25 hPa. L'espace aérien est ainsi découpé en tranches horizontales appelées niveaux de vol et désignées par les lettres " FL " (flight level) suivies de l'altitude en centaines de pieds : " FL310 " désigne une altitude de croisière de 31000 pieds au-dessus de la surface isobare de référence.

Eurocontrol est l'organisation européenne chargée de la navigation aérienne, elle gère plusieurs dizaines de milliers de vol par jour. Toute compagnie qui souhaite faire traverser le ciel européen à un de ses avions doit soumettre à cet organisme un plan de vol comprenant un certain nombre d'informations : trajet, heure de départ, niveau de vol souhaité, etc. Muni de ces informations, Eurocontrol peut prévoir les secteurs aériens qui vont être surchargés et prendre des mesures en conséquence pour les désengorger : retard au décollage, modification de la route à suivre, etc.

Nous modélisons (de manière très simplifée) les plans de vol gérés par Eurocontrol sous la forme d'une base de données comportant deux tables :

la table `vol` qui répertorie les plans de vol déposés par les compagnies aériennes ; elle contient les colonnes

- `id_vol` : numéro du vol (chaine de caractères) ;
- `depart` : code de l'aéroport de départ (chaine de caractère) ;
- `arrivee` : code de l'aéroport d'arrivée (chaine de caractère) ;
- `jour` : jour du vol (de type `date`, au format `aaaa-mm-jj` ;
- `heure` : heure de décollage souhaitée (de type `time`, au format `hh:mm`) ;
- `niveau` : niveau de vol souhaité (entier).

`id_vol`	`depart`	`arrivee`	`jour`	`heure`	`niveau`
AF1204	CDG	FCO	2016-05-02	07 :35	300
AF1205	FCO	CDG	2016-05-02	10 :25	300
AF1504	CDG	FCO	2016-05-02	10 :05	310
AF1505	FCO	CDG	2016-05-02	13 :00	310

Figure 1 extrait de la table `vol` : vols de la compagnie Air France entre les aéroports Charles-de-Gaulle (Paris) et Léonard-de-Vinci à Fiumicino (Rome)

la table `aeroport` qui répertorie les aéroports européens ; elle contient les colonnes

- `id_aero` : code de l'aéroport (chaine de caractères) ;
- `ville` : principale ville desservie (chaine de caractères) ;
- `pays` : pays dans lequel se situe l'aéroport (chaine de caractères).

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie

Figure 2 extrait de la table `aeroport`

Les types SQL date et time permettent de mémoriser respectivement un jour du calendrier grégorien et une heure du jour. Deux valeurs de type date ou de type time peuvent être comparées avec les opérateurs habituels (=, <, <=, etc.). La comparaison s'effectue suivant l'ordre chronologique. Ces valeurs peuvent également être comparées à une chaine de caractères correspondant à leur représentation externe ('aaaa-mm-jj' ou 'hh :mm').

1. Écrire une requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi.
2. Écrire une requête SQL qui fournit la liste des numéros de vols au départ d'un aéroport desservant Paris le 2 mai 2016.
3. Que fait la requête suivante ?

```
SELECT id_vol
FROM vol
    JOIN aeroport AS d ON d.id_aero = depart
    JOIN aeroport AS a ON a.id_aero = arrivee
WHERE
    d.pays = 'France' AND
    a.pays = 'France' AND
    jour = '2016-05-02'
```

4. Certains vols peuvent engendrer des conflits potentiels : c'est par exemple le cas lorsque deux avions suivent un même trajet, en sens inverse, le même jour et à un même niveau. Écrire une requête SQL qui fournit la liste des couples (Id_1, Id_2) des identifiants des vols dans cette situation.

Exercice 3

(d'après Mines-Ponts 2016.) Pour suivre la propagation des épidémies, de nombreuses données sont recueillies par les institutions internationales comme l'O.M.S. Par exemple, pour le paludisme, on dispose de deux tables :

– la table `palu` recense le nombre de nouveaux cas confirmés et le nombre de décès liés au paludisme ; certaines lignes de cette table sont données en exemple (on précise que `iso` est un identifiant unique pour chaque pays) :

nom	iso	annee	cas	deces
Bresil	BR	2009	309 316	85
Bresil	BR	2010	334 667	76
Kenya	KE	2010	898 531	26 017
Mali	ML	2011	307 035	2 128
Ouganda	UG	2010	1 581 160	8431

– la table `demographie` recense la population totale de chaque pays ; certaines lignes de cette table sont données en exemple :

pays	periode	pop
BR	2009	193 020 000
BR	2010	194 946 000
KE	2010	40 909 000
ML	2011	14 417 000
UG	2010	33 987 000
...		

1. Au vu des données présentées dans la table `palu`, parmi les attributs `nom`, `iso` et `annee`, quels attributs peuvent servir de clé primaire ? Un couple d'attributs pourrait-il servir de clé primaire ? (on considère qu'une clé primaire peut posséder plusieurs attributs). Si oui, en préciser un.
2. Écrire une requête en langage SQL qui récupère depuis la table `palu` toutes les données de l'année 2010 qui correspondent à des pays où le nombre de décès dus au paludisme est supérieur ou égal à 1 000.

On appelle taux d'incidence d'une épidémie le rapport du nombre de nouveaux cas pendant une période donnée sur la taille de la population-cible pendant la même période. Il s'exprime généralement en " nombre de nouveaux cas pour 100 000 personnes par année ". Il s'agit d'un des critères les plus importants pour évaluer la fréquence et la vitesse d'apparition d'une épidémie.

3. Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `palu`.
4. Écrire une requête en langage SQL permettant de déterminer le nom du pays ayant eu le deuxième plus grand nombre de nouveaux cas de palu-

disme en 2010 (on pourra supposer qu'il n'y a pas de pays ex-æquo pour les nombres de cas).

Exercice 4

Charger la base de donnée `movie.sqlite`. Elle contient 3 tables vérifiant les schémas suivants :

```
actor ( id INTEGER, name VARCHAR(35))
casting (movieid INTEGER, actorid INTEGER, ord INTEGER)
movie (id INTEGER, title VARCHAR(70), yr DECIMAL(4), score
FLOAT,
        votes INTEGER, director INTEGER)
```

Elle porte sur tous les films de cinema, leur casting, acteurs et réalisateurs jusqu'à l'année 2000. La table `actor` contient les acteurs mais aussi les réalisateurs

1. Obtenir titre et score des films ayant obtenu au moins 10000 votes classés par score décroissant
2. Obtenir les titres de tous les films réalisés par Alfred Hitchcock.
3. Obtenir les noms de tous les acteurs et actrices ayant tourné dans un film d'Alfred Hitchcock.
4. Obtenir nom et notes moyennes de tous les réalisateurs classés par score moyen de leur film décroissant.
5. Obtenir le titre du film ayant le meilleur score, puis le casting de ce film.

Exercice 5

La base de données `geographie.sqlite` contient trois tables :

• La table `communes` contient toutes les communes de France. Ses attributs sont :

- `num_departement` : le numéro du département,
- `nom` : le nom de la commune,
- `canton`,
- `population_2010` : le nombre d'habitants lors du recensement de 2010,
- `population_1999` : le nombre d'habitants lors du recensement de 1999,
- `surface` : la surface de la commune en km^2,
- `longitude` : la longitude du centre de la commune,
- `latitude` : la latitude du centre de la commune,
- `zmin` : l'altitude minimale de la commune,
- `zmax` : l'altitude maximale de la commune,

• La table `departements` contient tous les départements de France. Ses attributs sont :

- `num_departement` : le numéro du département,
- `num_region` : le numéro de la région,
- `nom` : le nom du département,

• La table `regions` contient toutes les régions de France.
Ses attributs sont :

- `num_region` : le numéro de la région,
- `nom` : le nom de la région,

Dans cette partie on travaillera sous `python` avec le module `sqlite3`. Les graphiques utiliseront la fonction `scatter()` du module `matplotlib.pyplot` :

Si `LX` et `LY` sont deux listes/tableaux de nombres de même longueur N, l'instruction : `plt.scatter(LX,LY)` représentera tous les points du plan d'abscisses respectives dans `LX` et d'ordonnées respectives dans `LY` par de petits ronds pleins (des disques).

Si `Aires` et `Couleurs` sont deux tableaux de nombres (de longueurs N), alors avec les options :

```
plt.scatter(LX, LY, s = Aires, c= Couleurs)
```

les disques seront chacun dotés d'une aire et d'une couleur.
On peut effectuer des requêtes `sqlite` sous `python` en utilisant le module `sqlite3`. C'est particulièrement pratique pour tracer des graphiques à partir de données obtenues d'une base de donnée SQLite.

•Création/connection d'une BdD :

```
import sqlite3

# Connection à une bdd (ou création d'une nouvelle bdd):
bdd = sqlite3.connect('ma_Base_De_Donnee.sqlite')
```

```
# Ou création dans la RAM :
bdd = sqlite3.connect(':memory:')
```

•Requêtes SQL :

```
b = bdd.cursor()        # Création d'un 'curseur'
b.execute("SELECT * FROM ma_table WHERE nom ='dubois' ")
# requête SQL
b.fetchone()            # retourne UN résultat

b.execute("SELECT * FROM ma_table")
b.fetchall()        # Retourne la liste de TOUS les résultats.
```

•Exécution d'un script SQL (suite de requêtes) :

```
b.executescript(""" SELECT * FROM ma_table WHERE nom = 'dupond';
                                etc...
        SELECT * FROM ma_table WHERE nom = 'durand'; """)
```

•Pour une exécution automatique :

```
b.execute("CREATE TABLE table2 (prenom TEXT, nom TEXT, age INT)")
enr = [ ('paul', 'durand', 23), ('andre', 'durand', 44)]
b.executemany("INSERT INTO table2 VALUES (?, ?, ?)", enr)
```

•Pour enregistrer les modifications apportées :

```
# Première solution :
bdd.commit()

# Deuxième solution : insérer les requêtes dans :
with bdd:
    b.executemany("INSERT INTO table2 VALUES (?, ?, ?)", enr)
    etc ...
```

•Fermeture de la base :

```
bdd.close()
```

1. Tracer la carte des communes de métropole (département $\leqslant 95$).

2. Tracer la carte des 100 communes de métropole les plus peuplées. La taille de chacune sera proportionnelle à sa population

3. Tracer la carte des communes de métropoles, colorées selon leur altitude.